

Efficient Pool-Based Active Learning of Halfspaces

Alon Gonen `alongnn@cs.huji.ac.il` Sivan Sabato `sabato@cs.huji.ac.il`

Shai Shalev-Shwartz `shais@cs.huji.ac.il`

Benin school of CSE
The Hebrew University
Givat Ram, Jerusalem 91904, Israel

Abstract

We study pool-based active learning of halfspaces, in which a learner receives a pool of unlabeled examples, and iteratively queries a teacher for the labels of examples from the pool, in order to identify all the labels of pool examples. We revisit the idea of greedily selecting examples to label, and use it to derive an efficient algorithm, called ALuMA, that approximates the optimal label complexity for a given pool in \mathbb{R}^d . We show that ALuMA obtains an $O(d^2 \log(d))$ approximation factor if the examples in the pool are numbers with a finite accuracy. We further prove a result for general hypothesis classes, showing that a slight change to the greedy approach leads to an improved target-dependent guarantee on the label complexity. In particular, we conclude a better guarantee for ALuMA if the target hypothesis has a large margin. We further compare our approach to other common active learning strategies, and provide a theoretical and empirical evaluation of the advantages and disadvantages of the approach.

1 Introduction

Pool-based active learning [McCallum and Nigam, 1998] is useful in many data-laden applications, where unlabeled data is abundant but labeling is expensive. In this setting the learner receives as input a set of instances, denoted $X = \{x_1, \dots, x_m\}$. Each instance x_i is associated with a label $L(i)$, which is initially unknown to the learner. The learner has access to a teacher, represented by the oracle $L : [m] \rightarrow \{-1, 1\}$. The goal of the learner is to find the values $L(1), \dots, L(m)$ using as few calls to L as possible. We assume that L is determined by a function h taken from a pre-defined hypothesis class \mathcal{H} . That is, $\exists h \in \mathcal{H}$ such that for all i , $L(i) = h(x_i)$. We denote this by $L \Leftarrow h$. A pool-based algorithm can be used to learn a classifier in the standard PAC model, while querying fewer labels. We discuss this in Section 6. We mainly deal with the hypothesis class of homogeneous halfspaces in \mathbb{R}^d , namely, $X \subset \mathbb{R}^d$ and $\mathcal{H} = \{x \mapsto \text{sgn}(\langle w, x \rangle) : w \in \mathbb{R}^d\}$, where $\langle w, x \rangle$ is the inner product between the vectors w and x .

The set of all hypotheses in \mathcal{H} that are consistent with the labels that currently known to the learner is called the *version space*. Many active learning algorithms maintain a version space, and use it to decide which label to query next. For example, the CAL algorithm [Cohn et al., 1994] selects an instance at random and queries its label only if there are two hypotheses in the version space that disagree on its label. Tong and Koller [2002] proposed a more aggressive greedy selection policy for halfspaces: query the instance from the pool that splits the version space as evenly as possible, in terms of volume in \mathbb{R}^d . To implement this policy, one would need to calculate the volume of a convex body (the version space), which is known to be computationally intractable. Tong and Koller implemented several heuristics that attempt to follow their proposed selection principle using an efficient algorithm. For instance, they suggest to choose the example which is closest to the max-margin solution of the data labeled so far. However, none of their heuristics provably follow this

greedy selection policy. Our first contribution is an efficient algorithm, which relies on randomized approximation of the volume of the version space [e.g. Kannan et al., 1997]. This allows us to prove that our algorithm follows an *approximate* greedy rule for minimizing the volume of the version space.

The proposed algorithm can be theoretically analyzed as follows. Given a pool-based active learning algorithm \mathcal{A} , denote by $N(\mathcal{A}, h)$ the number of calls to L that \mathcal{A} makes before outputting $(L(x_1), \dots, L(x_m))$, for $L \in h$. The *worst-case label complexity* of \mathcal{A} is defined to be $\max_{h \in \mathcal{H}} N(\mathcal{A}, h)$ and the *average-case label complexity* of \mathcal{A} is defined to be $\mathbb{E}_{h \sim P} N(\mathcal{A}, h)$, where P is some predefined distribution over the hypothesis class \mathcal{H} . We denote the optimal worst-case label complexity for the given pool by OPT_{\max} and the optimal average label complexity (for some fixed distribution over hypotheses) by OPT_{avg} .

Dasgupta [2005] showed that if an exact greedy algorithm splits the probability mass of the version space (as defined by P) as evenly as possible, then its average label complexity using the same P is bounded by $O(\log(1/p_{\min}) \cdot \text{OPT}_{\text{avg}})$, where $p_{\min} = \min_{h \in \mathcal{H}} P(h)$. Golovin and Krause [2010] extended Dasgupta’s result and showed that a similar bound holds for an approximate greedy rule. They also showed that the worst-case label complexity of an approximate greedy rule is at most $O(\log(1/p_{\min}) \cdot \text{OPT}_{\max})$, thus extending a result of Arkin et al. [1993].

The distribution P over hypotheses that matches our volume-splitting strategy is one that draws a halfspace uniformly from the unit ball in \mathbb{R}^d .¹ In this case $P(h)$ is the probability mass of all the hypotheses inducing the same labeling as h on X . If there are instances in X that are very close to each other, then p_{\min} might be very small. Our second contribution is to show that mild conditions suffice to guarantee that p_{\min} is bounded from below. In particular, by proving a variant of a result due to Muroga et al. [1961], we show that if the examples in the pool X are stored using number of a finite accuracy $1/c$, then $p_{\min} \geq (c/d)^{d^2}$. It follows that the worst-case label complexity of our algorithm is at most $O(d^2 \log(d/c)) \cdot \text{OPT}_{\max}$.

While this result provides us with a uniform lower bound on p_{\min} , in many real-world situations the probability of the target hypothesis (i.e., one that is consistent with L) could be much larger than p_{\min} . A noteworthy example is when the target hypothesis separates X with a margin of γ . In this case, it can be shown that the probability of the target hypothesis is at least γ^d , which can be significantly larger than p_{\min} . An immediate question is therefore: can we obtain a *target-dependent* label complexity bound of $\log(1/P(h)) \cdot \text{OPT}_{\max}$, where h is the target hypothesis?

We prove that such a target dependent bound *does not* hold for a general approximate greedy algorithm. Nonetheless, in our third main contribution we show that by introducing an algorithmic change to the approximate greedy policy, we do obtain the label complexity bound of $\log(1/P(h)) \cdot \text{OPT}_{\max}$. In particular, we run an approximate greedy procedure, but stop the procedure early, before reaching a pure version space that exactly matches the labeling of the pool. We then use an approximate majority vote over the version space to determine the labels of X .

We also derive lower bounds, showing that the dependence of our label-complexity guarantee on the accuracy c , or the margin parameter γ , is indeed necessary and is not an artifact of our analysis. We do not know if the dependence of our bounds on d is tight. It should be noted that some of the most popular learning algorithms (e.g. SVM, Perceptron, and AdaBoost) rely on a large-margin assumption to derive dimension-independent sample complexity guarantees. In contrast, here we use the margin for computational reasons. Our approximation guarantee depends logarithmically on the margin parameter, while the sample complexities of SVM, Perceptron, and AdaBoost depend polynomially on the margin. Hence, we require a much smaller margin than these algorithms do. Balcan et al. [2007] proposed an active learning algorithm with dimension-independent guarantees under a margin assumption. These guarantees hold for a restricted class of data distributions.

Lastly, we compare the greedy approach of our algorithm to other previously proposed active learning strategies, both theoretically and experimentally. We underscore cases in which it can be beneficial to apply an aggressive greedy approach instead of a more mellow selective-sampling approach, but also show that each approach has advantages and disadvantages in different cases. The empirical evaluation indicates that our algorithm, which can be implemented in practice, achieves state-of-the-art results. It further suggests that aggressive approaches can be better than mellow approaches in

¹We discuss the challenges presented by other natural choices of P in Section 5

some practical settings as well. In this work we do not treat the case of labeling errors—this is left for future work.

The rest of the paper is organized as follows. We start in Section 2 with a formal statement of our main results, and present our algorithm and its analysis in Section 3. We give a proof sketch of our main theorem in Section 4, Proofs for the rest of the results are deferred to the appendix. In Section 5 we consider other possible solutions to the problem of efficient greedy learning of halfspaces, and discuss the challenges they present. Our comparison to other algorithms is given in Section 6.

2 Main Results

Let P be a distribution over a hypothesis class \mathcal{H} . Given the pool $X = \{x_1, \dots, x_m\}$, and some $h \in \mathcal{H}$, denote by $P(h)$ the probability mass of the set $\{h' \in \mathcal{H} : \forall i, h'(x_i) = h(x_i)\}$. Let $V_t \subseteq \mathcal{H}$ be the version space of an active learner after t queries. For a given pool example $x \in X$, denote by $V_{t,x}^j$ the version spaces that would result if the algorithm now queried x and received label j . An algorithm \mathcal{A} is called α -approximately greedy with respect to P , for $\alpha \geq 1$, if at each iteration $t = 1, \dots, T$, the pool example x that \mathcal{A} decides to query satisfies

$$P(V_{t,x}^1)P(V_{t,x}^{-1}) \geq \frac{1}{\alpha} \max_{\tilde{x} \in X} P(V_{t,\tilde{x}}^1)P(V_{t,\tilde{x}}^{-1}),$$

and the algorithm's output is $(h(x_1), \dots, h(x_m))$ for some $h \in V_T$. An algorithm is exactly greedy if it is approximately greedy with $\alpha = 1$.

We say that \mathcal{A} outputs an approximate majority vote if whenever V_T is “pure” enough, the algorithm outputs the majority vote on V_T . Formally, \mathcal{A} outputs a β -approximate majority for $\beta \in (\frac{1}{2}, 1)$ if whenever there exists a labeling $Z : X \rightarrow \{\pm 1\}$ such that $P_{h \sim P}[Z \neq h \mid h \in V_T] \geq \beta$, \mathcal{A} outputs Z . In the following theorem we provide a target-dependent label complexity bound, which holds for any approximate greedy algorithm that outputs an approximate majority vote.

Theorem 1 *Let $X = \{x_1, \dots, x_m\}$. Let \mathcal{H} be a hypothesis class, and let P be a distribution over \mathcal{H} . Suppose that \mathcal{A} is α -approximately greedy with respect to P . Further suppose that it outputs a β -approximate majority vote. If \mathcal{A} is executed with input (X, L, T) where $L \neq h \in \mathcal{H}$ and $T \geq \alpha(2 \ln(1/P(h)) + \ln(\frac{\beta}{1-\beta})) \cdot \text{OPT}_{\max}$, then \mathcal{A} outputs $L(1), \dots, L(m)$.*

Denote $p_{\min} = \min_{h \in \mathcal{H}} P(h)$. When $P(h) \gg p_{\min}$, the bound in Theorem 1 is stronger than the guarantee $\forall h \in \mathcal{H}, N(\mathcal{A}, h) \leq O(\log(1/p_{\min}) \cdot \text{OPT}_{\max})$, obtained by Golovin and Krause [2010]. Importantly, the following theorem shows that this improved bound *cannot be obtained* for a general approximate-greedy algorithm, even in a simple case such as the problem of *thresholds on the line*. In this setting, the examples are in $[0, 1]$, and the hypothesis class includes all the hypotheses defined by a threshold on $[0, 1]$. Formally $\mathcal{H}_{\text{line}} = \{h_c \mid c \in [0, 1], h_c(x) = 1 \Leftrightarrow x \geq c\}$.²

Theorem 2 *Consider pool-based active learning on $\mathcal{H}_{\text{line}}$, and assume that P on $\mathcal{H}_{\text{line}}$ selects h_c by drawing the value c uniformly from $[0, 1]$. For any $\alpha > 1$ there exists an α -approximately greedy algorithm \mathcal{A} such that for any $m > 0$ there exists a pool $X \subseteq [0, 1]$ of size m , and a threshold c such that $P(h_c) = 1/2$, while the label-complexity of \mathcal{A} for $L \neq h_c$ is $\frac{m}{\log(m)} \cdot \text{OPT}_{\max}$.*

Interestingly, this theorem does not hold for $\alpha = 1$, that is for the exact greedy algorithm. This follows from Theorem 7, which we state and prove in Section 6.

So far we have considered a general hypothesis class. We now discuss the class of halfspaces \mathcal{W} . Recall that each halfspace can be described using a vector w from the unit ball of \mathbb{R}^d . For simplicity, we will slightly overload notation and sometimes use w to denote the halfspace it determines. We let P be the distribution that selects a vector w uniformly from the unit ball in \mathbb{R}^d . Our algorithm for halfspaces, which is called ALuMA, is described in Section 3. ALuMA receives as input an extra parameter $\delta \in (0, 1)$, which serves as a measure of the desired confidence level. The following lemma shows that ALuMA has the desired properties described above with high probability.³

²This setting is isomorphic to the case of homogeneous classifiers with examples on a line in \mathbb{R}^2 .

³The definition of OPT_{\max} requires an algorithm that always succeeds, while we allow ALuMA to fail with small probability. This restriction on OPT_{\max} is made for convenience; The same approximation factor can be achieved when the optimal algorithm is allowed the same randomization power as ALuMA—See Appendix B.

Lemma 3 *If ALuMA is executed with confidence δ , then with probability $1 - \delta$ over its internal randomization, ALuMA is 4-approximately greedy and outputs a 2/3-approximate majority vote. Furthermore, ALuMA is polynomial in the pool size, the dimension, and $\log(1/\delta)$.*

Combining the above lemma with Theorem 1 we immediately obtain that ALuMA's label complexity is $O(\log(1/P(h)) \cdot \text{OPT}_{\max})$. We can upper-bound $\log(1/P(h))$ using the familiar notion of *margin*: For any hypothesis $h \in \mathcal{W}$ defined by some $w \in \mathbb{R}^d$, let $\gamma(h)$ be the maximal margin of the labeling of X by h , namely $\gamma(h) = \max_{v: \|v\|=1} \min_{i \in [m]} h(x_i) \langle v, x_i \rangle$. It is possible to show (see Lemma 9 in Appendix A.2) that $P(h) \geq \Omega(\gamma(h)^d)$. As a corollary we obtain:

Theorem 4 *Let $X = \{x_1, \dots, x_m\} \subseteq \mathbb{B}_1^d$, where \mathbb{B}_1^d is the unit Euclidean ball of \mathbb{R}^d . Let $\delta \in (0, 1)$ be a confidence parameter. Suppose that ALuMA is executed with input (X, L, T, δ) , where $L \Leftarrow h \in \mathcal{W}$ and $T \geq 4(2d \ln(2/\gamma(h)) + \ln(2)) \cdot \text{OPT}_{\max}$. Then, with probability of at least $1 - \delta$ over ALuMA's own randomization, it outputs $L(1), \dots, L(m)$.*

We can consider the minimal possible margin, $\gamma = \min_{h \in \mathcal{W}} \gamma(h)$, and deduce from Theorem 4, or from the results of Golovin and Krause [2010], a uniform approximation factor of $O(d \log(1/\gamma))$. How small can γ be? The following result bounds this minimal margin from below under the reasonable assumption that the examples are represented by numbers of a finite accuracy.

Lemma 5 *Let $c > 0$ be such that $1/c$ is an integer and suppose that $X \subset \{-1, -1 + c, \dots, 1 - c, 1\}^d$. Then, $\min_{h \in \mathcal{W}} \gamma(h) \geq (c/\sqrt{d})^{d+2}$.*

The proof, given in Appendix A.3, is an adaptation of a classic result due to Muroga et al. [1961]. We conclude that $p_{\min} = \Omega((c/d)^{d^2})$, and deduce an approximation factor of $d^2 \log(d/c)$ for the worst-case label complexity of ALuMA. The exponential dependence of the minimal margin on d here is necessary: As shown in Håstad [1994], the minimal margin can indeed be exponentially small, even if the points are taken only from $\{\pm 1\}^d$.

We also derive a lower bound, showing that the dependence of our bounds on γ or on c is necessary. Whether the dependence on d is also necessary is an open question for future work.

Theorem 6 *Fix $\alpha \geq 1$. For any $c > 0$ with $1/c$ an integer, there exists a pool from $\{-1, -1 + c, \dots, 1 - c, 1\}^2$, for which $\min_{h \in \mathcal{W}} \gamma(h) = c/2$, and any α -approximately greedy algorithm that outputs a majority vote will require $\Omega\left(\frac{\log(1/c)}{\log \log(1/c)} \text{OPT}_{\max}\right)$ labels.*

3 The ALuMA algorithm

We now describe our algorithm, listed below as Alg. 1, and explain why Lemma 3 holds. We name the algorithm *Active Learning under a Margin Assumption* or ALuMA. Its inputs are the unlabeled sample X , the labeling oracle L , the maximal allowed number of label queries T , and the desired confidence $\delta \in (0, 1)$. It returns the labels of all the examples in X . Two building blocks are required in order to implement an algorithm with the desired guarantees for halfspaces. First, we need to be able to select a pool-example that approximately maximizes $P(V_{t,x}^1) \cdot P(V_{t,x}^{-1})$. Second, we need to be able to output the majority vote of a version space that has a high enough purity level.

For the first building-block, we need to calculate the volumes of the sets $V_{t,x}^1$ and $V_{t,x}^{-1}$. Both of these sets are convex sets obtained by intersecting the unit ball with halfspaces. The problem of calculating the volume of such convex sets in \mathbb{R}^d is #P-hard if d is not fixed [Brightwell and Winkler, 1991]. Moreover, deterministically approximating the volume is NP-hard in the general case [Matoušek, 2002]. Luckily, it is possible to approximate this volume using randomization. Specifically, in Kannan et al. [1997] a randomized algorithm is provided such that for any convex body $K \subseteq \mathbb{R}^d$ with an efficient separation oracle, with probability at least $1 - \delta$ the algorithm returns a non-negative number Γ such that $(1 - \epsilon)\Gamma < P(K) < (1 + \epsilon)\Gamma$. The algorithm is polynomial in $d, 1/\epsilon, \ln(1/\delta)$. ALuMA uses this algorithm to estimate $P(V_{t,x}^1)$ and $P(V_{t,x}^{-1})$ with sufficient accuracy. Using the guarantees of this algorithm and the constants in ALuMA, we can show that ALuMA is 4-approximately greedy with probability $1 - \delta/2$. We denote an execution of this algorithm on a convex body K by $\Gamma \leftarrow \text{VolEst}(K, \epsilon, \delta)$.

Algorithm 1 The ALuMA algorithm

```

1: Input:  $X = \{x_1, \dots, x_m\}$ ,  $L : [m] \rightarrow \{-1, 1\}$ ,  $T, \delta$ 
2:  $I_1 \leftarrow [m]$ ,  $V_1 \leftarrow \mathbb{B}_1^d$ 
3: for  $t = 1$  to  $T$  do
4:    $\forall i \in I_t, j \in \{\pm 1\}$ , do  $\hat{v}_{x_i, j} \leftarrow \text{VolEst}(V_{t, x_i}^j, \frac{1}{3}, \frac{\delta}{4mT})$ 
5:   Select  $i_t \in \text{argmax}_{i \in I_t} (\hat{v}_{x_i, 1} \cdot \hat{v}_{x_i, -1})$ 
6:    $I_{t+1} \leftarrow I_t \setminus \{i_t\}$ 
7:   Request  $y = L(i_t)$ 
8:    $V_{t+1} \leftarrow V_t \cap \{w : y \langle w, x_{i_t} \rangle > 0\}$ 
9: end for
10:  $M \leftarrow \lceil 72 \ln(2/\delta) \rceil$ .
11: Draw  $w_1, \dots, w_M$   $\frac{1}{M}$ -uniformly from  $V_{T+1}$ .
12: For each  $x_i$  return the label  $y_i = \text{sgn} \left( \sum_{j=1}^M \text{sgn}(\langle w_j, x_i \rangle) \right)$ .
```

To output an approximate majority vote from the final version space V , we would like to uniformly draw several hypotheses from V and label X according to a majority vote over these hypotheses. We can efficiently draw a hypothesis *approximately* uniformly from V , by using the hit-and-run algorithm [Lovász, 1999]. For a convex body K , the hit-and-run algorithm runs $\tilde{O}(d^3/\lambda^2)$ steps and returns a random sample according to a distribution which is λ -close in total-variation distance to the uniform distribution over K . Using the constants in ALuMA, we can show that ALuMA outputs a $2/3$ -approximate majority with probability $1 - \delta/2$.

4 Proof Sketch for Theorem 1

We give here the main steps in the proof of Theorem 1. Fix a pool X . For any algorithm alg , denote by $V_t(\text{alg}, h)$ the version space induced by the first n labels it queries if the true labeling of the pool is consistent with h . Denote the average version space reduction of alg after n queries by

$$f_{\text{avg}}(\text{alg}, n) = 1 - \mathbb{E}_{h \sim P}[P(V_n(\text{alg}, h))].$$

Golovin and Krause [2010] prove that since \mathcal{A} is α -approximately greedy, for any pool-based algorithm alg ,

$$f_{\text{avg}}(\mathcal{A}, n) \geq f_{\text{avg}}(\text{alg}, k) - \exp(-n/\alpha k). \quad (1)$$

Let opt be an algorithm that achieves OPT_{\max} . It can be shown that for any hypothesis $h \in \mathcal{H}$ and any active learner alg ,

$$f_{\text{avg}}(\text{opt}, \text{OPT}_{\max}) - f_{\text{avg}}(\text{alg}, n) \geq P(h)(P(V_n(\text{alg}, h)) - P(h)).$$

Combining this with Equation (1) we conclude that if \mathcal{A} is α -approximately greedy then

$$\frac{P(h)}{P(V_n(\mathcal{A}, h))} \geq \frac{P(h)^2}{\exp(-\frac{n}{\alpha \text{OPT}_{\max}}) + P(h)^2}.$$

This means that if $P(h)$ is large enough and we run an approximate greedy policy, then after a sufficient number of iterations, most of the remaining version space induces the correct labeling of the sample. Specifically, if $n \geq \alpha(2 \ln(1/P(h)) + \ln(\frac{\beta}{1-\beta})) \cdot \text{OPT}_{\max}$, then $P(h)/P(V_n(\mathcal{A}, h)) \geq \beta$. Since \mathcal{A} outputs a β -approximate majority labeling from $V_n(\mathcal{A}, h)$, \mathcal{A} returns the correct labeling.

5 On the difficulties in greedy active-learning for halfspaces

At first glance it might seem that there are simpler ways to implement an efficient greedy strategy for halfspaces, by using a different distribution P over the hypotheses. For instance, if there are m examples in d dimensions, Sauer's lemma states that the effective size of the hypothesis class of halfspaces will be at most m^d . One can thus use the uniform distribution over this finite class, and greedily reduce the number of possible hypotheses in the version space, obtaining a $d \log(m)$ factor relative to the optimal label complexity. However, a direct implementation of this method will be exponential in d , and it is not clear whether this approach has a polynomial implementation.

Another approach is to discretize the version space, by considering only halfspaces that can be represented as vectors on a d -dimensional grid $\{-1, -1 + c, \dots, 1 - c, 1\}^d$. This results in a finite hypothesis class of size $(2/c + 1)^d$, and we get an approximation factor of $O(d \log(1/c))$ for the greedy algorithm, compared to an optimal algorithm on the same finite class. However, it is not clear whether a greedy algorithm for reducing the number of such vectors in a version space can be implemented efficiently, since even determining whether a single grid point exists in a given version space is NP-hard [see e.g. Matoušek, 2002, Section 2.2].

Yet another possible direction for pool-based active learning is to greedily select a query whose answer would determine the labels of the largest amount of pool examples. The main challenge in this direction is how to analyze the label complexity of such an algorithm: it is unclear whether competitiveness with OPT can be guaranteed in this case. Investigating this idea, both theoretically and experimentally, is an important topic for future work. Note that the CAL algorithm Cohn et al. [1994], which we discuss in Section 6, can be seen as implementing a mellow version of this approach, since it decreases the so-called “disagreement region” in each iteration.

Inspecting our margin-dependent guarantees, one may wonder if a margin assumption alone can guarantee that OPT_{\max} is small. This is not the case, as evident by the following example, which is an adaptation of an example from Dasgupta [2005].

Example 1 Let $\gamma \in (0, \frac{1}{2})$ be a margin parameter. Consider a pool of m points in \mathbb{R}^d , such that all the points are on the unit sphere, and for each pair of points x_1 and x_2 , $\langle x_1, x_2 \rangle \leq 1 - 2\gamma$. It was shown in Shannon [1959] that for any $m \leq O(1/\gamma^d)$, there exists a set of points that satisfy the conditions above. For any point x in such a pool, there exists a (biased) halfspace that separates x from the rest of the points with a margin of γ . By adding a single dimension, this example can be transformed to one with homogeneous (unbiased) halfspaces. Each point in this pool can be separated from the rest of the points by a halfspace. Thus, if the correct labeling is all-positive, then all m examples need to be queried to label the pool correctly. Therefore $\text{OPT}_{\max} = m$.

6 Other Approaches: Theoretical and Empirical Comparison

We now compare the effectiveness of the approach implemented by ALuMA to other active learning strategies. ALuMA can be characterized by two properties: (1) its “objective” is to reduce the volume of the version space and (2) at each iteration, it aggressively selects an example from the pool so as to (approximately) minimize its objective as much as possible (in a greedy sense). We discuss the implications of these properties by comparing to other strategies. Property (1) is contrasted with strategies that focus on increasing the number of examples whose label is known. Property (2) is contrasted with strategies which are “mellow”, in that their criterion for querying examples is softer.

Much research has been devoted to the challenge of obtaining a substantial guaranteed improvement of label complexity over regular “passive” learning for halfspaces in \mathbb{R}^d . Examples (for the realizable case) include the Query By Committee (QBC) algorithm [Seung et al., 1992, Freund et al., 1997], the CAL algorithm [Cohn et al., 1994], and the Active Perceptron [Dasgupta et al., 2005]. These algorithms are not “pool-based” but rather use “selective-sampling”: they sample one example at each iteration, and immediately decide whether to ask for its label. Out of these algorithms, CAL is the most mellow, since it queries any example whose label is yet undetermined by the version space. Its “objective” can be described as reducing the number of examples which are labeled incorrectly, since it has been shown to do so in many cases [Hanneke, 2007, 2009, Friedman, 2009]. QBC and the active perceptron are less mellow. Their “objective” is similar to that of ALuMA since they decide on examples to query based on geometric considerations.

In the first part of our comparison, we discuss the theoretical advantages and disadvantages of different strategies, by considering some interesting cases from a theoretical perspective. In the second part we report an empirical comparison of several algorithms and discuss our conclusions.

6.1 Theoretical Comparison

The label complexity of the algorithms mentioned above is usually analyzed in the PAC setting, thus we translate our guarantees into the PAC setting as well for the sake of comparison. We define the (ϵ, m, D) -label complexity of an active learning algorithm to be the number of *label queries* that

are required in order to guarantee that given a sample of m unlabeled examples drawn from D , the error of the learned classifier will be at most ϵ (with probability of at least $1 - \delta$ over the choice of sample). A pool-based active learner can be used to learn a classifier in the PAC model by first sampling a pool of m unlabeled examples from D , then applying the pool-based active learner to this pool, and finally running a standard passive learner on the labeled pool to obtain a classifier. For the class of halfspaces, if we sample an unlabeled pool of $m = \tilde{\Omega}(d/\epsilon)$ examples, then the learned classifier will have an error of at most ϵ (with high probability over the choice of the pool).

To demonstrate the effect of the first property discussed above, consider again the simple case of thresholds on the line defined in Section 2. Compare two greedy pool-based active learners for $\mathcal{H}_{\text{line}}$: The first follows a binary search procedure, greedily selecting the example that increases the number of known labels the most. Such an algorithm requires $\log(m)$ queries to identify the correct labeling of the pool. The second algorithm queries the example that splits the version space as evenly as possible. Theorem 1 implies a label complexity of $O(\log(m) \log(1/\gamma(h)))$ for such an algorithm, since $\text{OPT}_{\text{max}} = \log(m)$. However, a better result holds for this simple case:

Theorem 7 *In the problem of thresholds on the line, for any pool with labeling L , the exact greedy algorithm requires at most $O(\log(1/\gamma(h)))$ labels. This is also the label complexity of any approximate greedy algorithm that outputs a majority vote.*

Comparing the $\log(m)$ guarantee of the first algorithm to the $\log(1/\gamma(h))$ guarantee of the second, we reach the (unsurprising) conclusion, that the first algorithm is preferable when the true labeling has a small margin, while the second is preferable when the true labeling has a large margin. This simple example accentuates the implications of selecting the volume of the version space as an objective. A similar implication can be derived by considering the PAC setting, replacing the binary-search algorithm with CAL, and letting $m = \tilde{\Theta}(1/\epsilon)$. On the single-dimensional line, CAL achieves a label-complexity of $O(\log(1/\epsilon)) = O(\log(m))$, similarly to the binary search strategy we described. Thus when ϵ is large compared to $\gamma(h)$, CAL is better than being greedy on the volume, and the opposite holds when the condition is reversed. QBC will behave similarly to ALuMA in this setting.

To demonstrate the effect of the second property described above—being aggressive versus being mellow, we consider the following example, adapted slightly from [Dasgupta, 2006].

Example 2 *Consider two circles parallel to the (x, y) plane in \mathbb{R}^3 , one at the origin and one slightly above it. For a given ϵ , fix $2/\epsilon$ points that are evenly distributed on the top circle, and $2/\epsilon$ points at the same angles on the bottom circle (see left illustration below). The distribution D_ϵ is an uneven mix of a uniform distribution over the points on the top circle and one over the points of the bottom circle: The top circle is given a much higher probability. All homogeneous separators label half of the bottom circle positively, but an unknown part of the top circle (see right illustration). The bottom points can be very helpful in finding the correct separator fast, but their probability is low.*



Dasgupta has demonstrated via this example that active learning can gain in label complexity from having significantly more unlabeled data. The following theorem shows that the aggressive strategy employed by ALuMA indeed achieves an exponential improvement when there are more unlabeled samples. In contrast, the mellow strategy of CAL does not significantly improve over passive learning in this case. We note that these results hold for any selective-sampling method that guarantees a similar error rate to passive ERM given the same sample size.

Theorem 8 *For all small enough $\epsilon \in (0, 1)$ there is a distribution D_ϵ of points in \mathbb{R}^3 , such that*

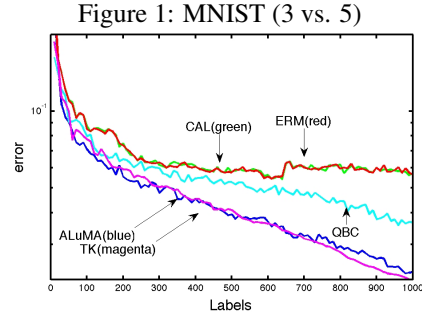
- *For $m = O(1/\epsilon)$, the $(\epsilon, m, D_\epsilon)$ -label complexity of any active learner is $\Omega(1/\epsilon)$.*
- *For $m = \Omega(\log^2(1/\epsilon)/\epsilon^2)$, the $(\epsilon, m, D_\epsilon)$ -label complexity of ALuMA is $O(\log^2(1/\epsilon))$.*
- *For any value of m , the $(\epsilon, m, D_\epsilon)$ -label complexity of CAL is $\Omega(1/\epsilon)$.*

In many applications, unlabeled examples are virtually free to sample, thus it can be worthwhile to allow the active learner to sample more examples than the passive sample complexity.⁴ We show in Appendix C another example in which the label complexity of CAL can be significantly worse than that of the optimal algorithm, even without more unlabeled examples. These examples strengthen the observation of Balcan et al. [2007] that in some cases a more aggressive approach is preferable.

On the other hand, CAL has a guaranteed label complexity for cases for which ALuMA currently has none. Its label complexity is bounded by $\tilde{O}(d\theta \log(1/\epsilon))$, where θ is the *disagreement coefficient*, a quantity that depends on the distribution and the target hypothesis [Hanneke, 2007, 2009]. Specifically, if D is uniform over a sphere centered at the origin, then for all target hypotheses $\theta = \Theta(\sqrt{d})$. Thus CAL achieves an exponential improvement over passive learning for this canonical example.

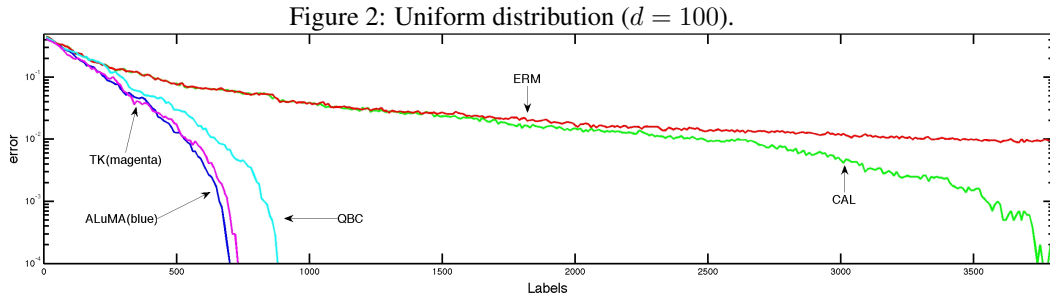
6.2 Empirical Comparison

We carried out an empirical comparison between the algorithms discussed above. Our goal is twofold: First, to evaluate ALuMA in practice, and second, to compare the performance of aggressive strategies compared to mellow strategies. The aggressive strategies are represented in this evaluation by ALuMA and one of the heuristics proposed by Tong and Koller [2002]. The mellow strategy is represented by CAL. QBC represents a middle-ground between aggressive and mellow. We also compare to a passive ERM algorithm—one that uses random labeled examples. We evaluated the algorithms over synthetic and real data sets and compared their label complexity performance. Details on our implementations and additional results are provided in Appendix D.



In the first experiment the data was digits 3 and 5 from the MNIST dataset. Figure 1 depicts the training error as a function of the label budget. It is striking to observe that CAL provides no improvement over passive ERM in the first 1000 examples, while this budget suffices to reach zero training error for ALuMA and TK. More results for MNIST and for another real data set can be found in Appendix D. In these experiments, the more aggressive learners consistently perform better.

The next experiment shows that ALuMA and TK outperform CAL and QBC even on a data sampled from the uniform distribution on a sphere in \mathbb{R}^d (see Figure 2). This result, and a similar one reported in the appendix, suggest that ALuMA might have a better guarantee than the general competitive result in case of the uniform distribution. This is an open question which is left for future work.



To summarize, in all of our tests, aggressive algorithms performed better than mellow ones. These results are not fully explained by current theory. The experiments also show that ALuMA and TK have comparable success in practice. One might hope that TK enjoys similar theoretical guarantees to those of ALuMA. In Appendix D we report a synthetic experiment that suggests the contrary.

⁴In the limit of an infinite number of unlabeled examples, if the distribution has a non-zero support on the entire domain, the pool-based setting becomes identical to the setting of membership queries [Angluin, 1988]. In contrast, we are interested in finite samples.

References

- D. Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.
- E.M. Arkin, H. Meijer, J.S.B. Mitchell, D. Rappaport, and S.S. Skiena. Decision trees for geometric models. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 369–378. ACM, 1993.
- M.F. Balcan, A. Broder, and T. Zhang. Margin based active learning. *Learning Theory*, pages 35–50, 2007.
- G. Brightwell and P. Winkler. Counting linear extensions is #p-complete. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC ’91, pages 175–181, 1991.
- D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- S. Dasgupta. Analysis of a greedy active learning strategy. *Advances in neural information processing systems*, 17:337–344, 2005.
- S. Dasgupta. Coarse sample complexity bounds for active learning. *Advances in neural information processing systems*, 18:235, 2006.
- S. Dasgupta, A. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. *Learning Theory*, pages 889–905, 2005.
- Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.
- E. Friedman. Active learning for smooth problems. In *Proceedings of the 22nd Conference on Learning Theory*, volume 1, pages 3–2, 2009.
- R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. *Advances in Neural Information Processing Systems (NIPS)*, 19, 2005.
- D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. In *Proceedings of International Conference on Learning Theory (COLT)*, 2010.
- S. Hanneke. Teaching dimension and the complexity of active learning. In *COLT*, 2007.
- S. Hanneke. Adaptive rates of convergence in active learning. In *COLT*, 2009.
- J. Håstad. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 7:484, 1994.
- R. Kannan, L. Lovász, and M. Simonovits. Random walks and an $o(n^5)$ volume algorithm for convex bodies. *Random structures and algorithms*, 11(1):1–50, 1997.
- L. Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86(3):443–461, 1999.
- J. Matoušek. *Lectures on discrete geometry*, volume 212. Springer Verlag, 2002.
- A. McCallum and K. Nigam. Employing em in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358, 1998.
- S. Muroga, I. Toda, and S. Takasu. Theory of majority decision elements. *Journal of the Franklin Institute*, 271(5):376–418, 1961.
- H.S. Seung, M. Oppor, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.
- C.E. Shannon. Probability of error for optimal codes in a gaussian channel. *Bell System Technical Journal*, 38:611–656, 1959.
- S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

Supplementary Material for NIPS Submission “Efficient Pool-Based Active Learning of Halfspaces”

A Proofs

A.1 Proof of Theorem 2

For the hypothesis class $\mathcal{H}_{\text{line}}$, the possible version spaces after a partial run of an active learner are all of the form $[a, b] \subseteq [0, 1]$. First, it is easy to see that binary search on the pool will identify any hypothesis in $[0, 1]$ using $\log(m)$ example, thus $\text{OPT}_{\text{max}} = \log(m)$.

Now, Consider an active learning algorithm that satisfies the following properties:

- If the current version space is $[a, b]$, it selects $x \in X$ such that $x = \min\{x \mid (x-a)(b-x) \geq \frac{1}{\alpha} \max_{\tilde{x} \in X \cap [a, b]} (\tilde{x} - a)(b - \tilde{x})\}$.
- When the budget of queries is exhausted, if the version space is $[a, b]$, label the points above a as positive and the rest as negative.

It is easy to see that this algorithm is α -approximately greedy, since in this problem $V_{t,x}^1 \cdot V_{t,x}^{-1} = (x-a)(b-x)$ for all $x \in [a, b] = V_t$. Now for a given pool size $m \geq 2$, consider a pool of examples defined as follows. First, let $x_1 = 1$, $x_2 = 1/2$ and $x_3 = 0$. Second, for each $i \geq 3$, define x_{i+1} recursively as the solution to $(x_{i+1} - x_i)(1 - x_{i+1}) = \frac{1}{\alpha}(1/2 - x_i)(1 - 1/2) = \frac{1}{2\alpha}(1/2 - x_i)$. Since $\alpha > 1$, it is easy to see by induction that for all $i \geq 3$, $x_{i+1} \in (x_i, \frac{1}{2})$. Furthermore, suppose the true labeling is induced by $h_{3/4}$; Thus the only pool example with a positive label is x_1 , and $P(h_{3/4}) = 1/2$. In this case, the algorithms we just defined will query all the pool examples x_4, x_5, \dots, x_m in order, and only then will it query x_2 and finally x_1 . If stopped at any time $t \leq m-1$, it will label all the points that it has not queried yet as positive, thus if $t < m-1$ the output will be an erroneous labeling. Finally, note that the same holds for the pool $x_1, x_2, x_4, \dots, x_m$ that does not include x_3 , so the algorithm must query this entire pool to identify the correct labeling.

A.2 Proof of lower bound for p_{\min}

Lemma 9 For all $h \in \mathcal{W}$, $P(h) \geq \left(\frac{\gamma(h)}{2}\right)^d$.

Proof Let $V = \{h' \in \mathcal{H} : \forall i, h'(x_i) = h(x_i)\}$. Choose $w \in \mathbb{B}_1^d$ such that $\forall x \in X$, $h(x)\langle w, x \rangle \geq \gamma$. For a given $v \in \mathbb{B}_1^d$, denote by $h_v \in \mathcal{H}$ the mapping $x \mapsto \text{sgn}(\langle v, x \rangle)$. Note that for all $v \in \mathbb{B}_1^d$ such that $\|w - v\| < \gamma$, $h_v \in V$. This is because for all $x \in X$,

$$\begin{aligned} h(x)\langle v, x \rangle &= \langle v - w, h(x) \cdot x \rangle + h(x)\langle w, x \rangle \\ &\geq -\|w - v\| \cdot \|h(x) \cdot x\| + \gamma > -\gamma + \gamma = 0, \end{aligned}$$

which implies $\text{sgn}(\langle v, x \rangle) = h(x)$. It follows that $\{v \mid h_v \in V\} \supseteq \mathbb{B}_1^d \cap B(w, \gamma)$, where $B(z, r)$ denotes the ball of radius r with center at z . Let $u = (1 - \gamma/2)w$. Then for any $z \in B(u, \gamma/2)$, we have $z \in \mathbb{B}_1^d$, since

$$\|z\| = \|z - u + u\| \leq \|z - u\| + \|u\| \leq \gamma/2 + 1 - \gamma/2 = 1.$$

In addition, $z \in B(w, \gamma)$ since

$$\|z - w\| = \|z - u + u - w\| \leq \|z - u\| + \|u - w\| \leq \gamma/2 + \gamma/2 = \gamma.$$

Therefore $B(u, \gamma/2) \subseteq \mathbb{B}_1^d \cap B(w, \gamma)$. We conclude that $\{v \mid h_v \in V\} \supseteq B(u, \gamma/2)$. Thus,

$$P(h) = \Pr_P[V] \geq \text{Vol}(B(u, \gamma/2)) / \text{Vol}(\mathbb{B}_1^d) \geq \left(\frac{\gamma}{2}\right)^d.$$

■

A.3 Proof of Lemma 5

Let us multiply all examples in the pool by $1/c$. Then, all the elements of all examples in the pool are integers. Choose a labeling L which is consistent with some w^* . Consider the optimization problem:

$$\min_w \|w\|^2 \text{ s.t. } \forall i, L(i)\langle w, x_i \rangle \geq 1.$$

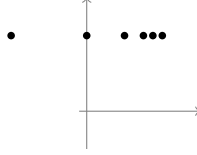
For simplicity assume that the pool of examples span all of \mathbb{R}^d . Then, it is easy to show that if w the solution to the above problem then there exist d linearly independent examples from the pool, denoted w.l.o.g. by x_1, \dots, x_d , such that $L(i)\langle w, x_i \rangle = 1$ for all i . In other words, w is the solution of the linear system $Aw = b$ where the rows of A are x_1, \dots, x_d and $b = (L(1), \dots, L(m))^T$.

By Cramer's rule, $w_i = \det(A_i)/\det(A)$, where A_i is obtained by replacing column i of A by the vector b . Since all elements of A are integers and A is invertible, we must have that $|\det(A)| \geq 1$. Therefore, $|w_i| \leq |\det(A_i)|$. Furthermore, by Hadamard's inequality, $|\det(A_i)|$ is upper bounded by the product of the norms of the columns of A_i . Since each element of A_i is upper bounded by $1/c$, we obtain that the norm of each column is at most $\frac{\sqrt{d}}{c}$, hence $|\det(A_i)| \leq (\sqrt{d}/c)^d$. It follows that $\|w\| \leq \sqrt{d}(\sqrt{d}/c)^d$. Hence, the margin is

$$\frac{1}{\|w\| \max_i \|x_i\|} \geq \frac{1}{\sqrt{d}(\sqrt{d}/c)^d \cdot \sqrt{d}/c} = \frac{1}{\sqrt{d}(\sqrt{d}/c)^{d+1}}.$$

A.4 Proof of Theorem 6

For any m , consider the pool $X = \{(-1, 1)\} \cup \{(1 - 2^{-k}, 1) : k = 0, 1, \dots, m-2\} \subset \mathbb{R}^2$, as illustrated below.



By following a binary search, the optimal algorithm can identify all the labels using $\log(m)$ queries. In contrast, it is easy to verify that the exact greedy algorithm will query all the points if all the examples but the rightmost are negative. We therefore obtain the approximation factor of $m/\log(m)$ on the label complexity compared to the optimal algorithm. Since the points lie on the grid with $c = 2^{-(m-2)}$ we obtain that the approximation factor is order of $\log(1/c)/\log \log(1/c)$. It is also easy to verify that the margin here is order of c . For an approximately-greedy strategy, the example can be adapted by replacing $1 - 2^{-k}$ with $1 - a^{-k}$ for $a \approx \alpha$, to make sure that the approximately-greedy strategy will query all the points in the same case. Thus in this case we get $m \approx \log(1/c)/\log(\alpha)$.

A.5 Proof of Theorem 7

First, assume that the algorithm is exactly greedy. A version space for $\mathcal{H}_{\text{line}}$ is described by a segment in $[a, b] \subseteq [0, 1]$, and a query at point α results in a new version space, $[a, \alpha]$ or $[\alpha, b]$, depending on the label. We now show that for every version space $[a, b]$, at most two greedy queries suffice to either reduce the size of the version space by a factor of at least $2/3$, or to determine the labels of all the points in the pool.

Assume for simplicity that the version space is $[0, 1]$, and denote the pool of examples in the version space by X . Assume w.l.o.g. that the greedy algorithm now queries $\alpha \leq \frac{1}{2}$. If $\alpha > 1/3$, then any answer to the query will reduce the version space size to less than $2/3$. Thus assume that $\alpha \leq 1/3$. If the query answer results in the version space $[0, \alpha]$ then we are done since this version space is smaller than $2/3$. We are left with the case that the version space after querying α is $[\alpha, 1]$. Since the algorithm is greedy, it follows that for $\beta = \min\{x \in X \mid x \geq \alpha\}$, we have $\beta \geq 1 - \alpha$: this is because if there was a point $\beta \in (\alpha, 1 - \alpha)$, it would cut the version space more evenly than α , in contradiction to the greedy choice of α . Note further that $(\alpha, 1 - \alpha)$ is larger than $[1 - \alpha, 1]$ since

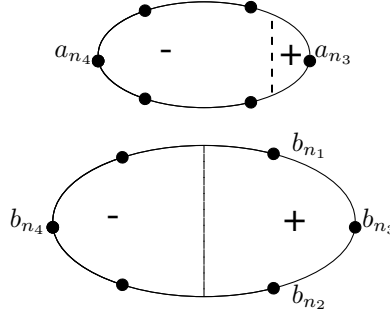


Figure 3: Illustration for the proof of Theorem 8.

$\alpha \leq 1/3$. Therefore, the most balanced choice for the greedy algorithm is β . If the query answer for β cuts the version space to $(\beta, 1]$ then we are done, since $1 - \beta \leq \alpha \leq 1/3$. Otherwise, the query answer leaves us with the version space (α, β) . This version space includes no more pool points, by the definition of β . Thus in this case the algorithm has determined the labels of all points.

It follows that if the algorithm runs at least t iterations, then the size of the version space after t iterations is at most $(2/3)^{t/2}$. If the true labeling has a margin of γ , we conclude that $(2/3)^{t/2} \geq \gamma$, thus $t \leq O(\log(1/\gamma))$.

A similar argument can be carried for ALuMA, using a smaller bound on α and more iterations due to the approximation, and noting that if the correct answer is in $(\alpha, 1 - \alpha)$ then a majority vote over thresholds drawn randomly from the version space will label the examples correctly.

A.6 Proof of Theorem 8

Proof Assume that $1/(2\epsilon)$ is an odd integer and $\epsilon < 1/8$. Let D_a be the uniform distribution over points on the top circle, defined by

$$S_a = \{a_n \stackrel{\text{def}}{=} (\frac{1}{\sqrt{2}} \cos 2\pi\epsilon n, \frac{1}{\sqrt{2}} \sin 2\pi\epsilon n, \frac{1}{\sqrt{2}}) : n \in \{0, 1, \dots, 1/\epsilon - 1\}\}.$$

Let D_b be the uniform distribution over points on the bottom circles, defined by

$$S_b = \{b_n \stackrel{\text{def}}{=} (\cos 2\pi\epsilon n, \sin 2\pi\epsilon n, 0) : n \in \{0, 1, \dots, 1/\epsilon - 1\}\}.$$

Let $D_{\epsilon/2}$ be the distribution $(1 - \tau)D_a + \tau D_b$, where $\tau = \frac{\epsilon}{4 \log(4/\epsilon)}$. Note that in order to label $D_{\epsilon/2}$ correctly with error no more than $\epsilon/2$, all the labels of points in S_a need to be determined. We prove each of the theorem statements in order. We consider the label complexity with high probability over the choice of unlabeled sample, where high probability is $1 - \delta$ for some fixed $\delta \in (0, 1/2)$.

Part I If the unlabeled sample contains only points from S_a , then an active learner has to query all the points in S_a to distinguish between a hypothesis that labels all of S_a positively and one that labels positively all but one point in S_a . Since the probability of the entire set S_b is $o(\epsilon)$, an i.i.d. sample of size $O(1/\epsilon)$, will not contain a point from S_b , thus any active learner will require $\Omega(1/\epsilon)$ labels.

Part II Assume now that the size of the sample is at least $\frac{4 \log(4/\epsilon) \log(1/(\epsilon\delta))}{\epsilon^2}$. It is easy to check that with probability at least $1 - \delta$, the sample contains all the points in $S_a \cup S_b$. Given such a sample as a pool, we now show that $\text{OPT}_{\max} = O(\log(1/\epsilon))$, by describing an active learning algorithm that achieves this label complexity:

1. For all possible separators, the points $b_0 = (1, 0, 0)$ and $b_{1/2\epsilon} = (-1, 0, 0)$ have different labels. The algorithm will first query these initial points, and then apply a binary search to find the boundary between negative and positive labels in S_b . This identifies the labels of all the points in S_b using $O(\log(1/\epsilon))$ queries.

2. Of the points in S_b , half are labeled positively and half negatively. Moreover, there are n_1, n_2 and $y \in \{-1, 1\}$ such that b_{n_1}, \dots, b_{n_2} are all labeled by y , and $n_2 - n_1 + 1 = |S_b|/2 = \frac{1}{2\epsilon}$ (see illustration in Figure 3). Let $n_3 = \frac{n_2 + n_1}{2}$ (this is the middle point with label y). n_3 is an integer because $n_2 - n_1$ is even, thus their sum is also even. Let $n_4 = \text{mod}(n_3 + 1/2\epsilon, 1/2\epsilon)$. Query the points a_{n_3} and a_{n_4} for their label.
3. If a_{n_3} and a_{n_4} each have a different label, apply a binary search starting from these points to find the boundaries between positive and negative labels in S_a , using $O(\log(1/\epsilon))$ queries. Otherwise, label all the examples in S_a by the label of a_{n_3} .

This algorithm uses $O(\log(1/\epsilon))$ queries to label the sample. If a_{n_3} and a_{n_4} have different labels, it is clear that the algorithm labels all the examples correctly. We only have left to prove that if they both have the same label, then all the examples in S_a also share that label. Let h^* be the true hypothesis, defined by some homogeneous separator, and assume w.l.o.g that $\{b_n \mid h^*(b_n) = 1\} = \{b_n \in S_b \mid b_n[1] > 0\}$ (note that no point has $b_n[1] = 0$ since $1/2\epsilon$ is odd). It follows that $n_3 = 0$ and $n_4 = 1/2\epsilon$, thus $a_{n_3} = (1/\sqrt{2}, 0, 1/\sqrt{2})$ and $a_{n_4} = (-1/\sqrt{2}, 0, 1/\sqrt{2})$ (see illustration in Figure 3). We prove the following lemma below:

Lemma 10 Assume $1/2\epsilon$ is odd. If $\{b_n \in S_b \mid h^*(b_n) = 1\} = \{b_n \mid b_n[1] > 0\}$ and $h^*(a_0) = h^*(a_{1/2\epsilon}) = y$ then $\forall a_n \in S_a, h^*(a_n) = y$.

It follows that $\text{OPT}_{\max} = O(\log(1/\epsilon))$.

To bound the label complexity of ALuMA, it suffices to bound from below the minimal margin of possible separators over the given sample. Let h^* be the correct hypothesis. By the same argument as in the proof of Lemma 5, there exists some $w \in \mathbb{R}^3$ that labels the sample identically to h^* and attains its maximal margin on three linearly independent points a, b, c from our sample. Hence, $Aw = \mathbf{1}$ where $A \in \mathbb{R}^{3 \times 3}$ is the matrix whose rows are $a, b, c \in S_a \cup S_b$. By Cramer's rule, for every $i \in [3]$

$$w[i] = \frac{\det A_i}{\det A},$$

where A_i is the matrix obtained from A by replacing the i^{th} column with the vector $\mathbf{1}$. Recall that the absolute value of the determinant of A is the volume of the parallelepiped whose sides are a, b and c . Since a, b, c are linearly independent, each of S_a and S_b includes at most two of them. Assume that $a, b \in S_a$ and $c \in S_b$. In this case, the surface area of the basis of this parallelepiped, defined by a and b , is at least $\frac{\sin 2\pi\epsilon}{\sqrt{2}}$, and the height is $1/\sqrt{2}$. Hence,

$$|\det A| \geq \frac{\sin 2\pi\epsilon}{2} = \Omega(\epsilon).$$

The case where two of the points are in S_b leads to an even larger lower bound. Since the elements in each A_i are in $[-1, 1]$, we also have that $|\det A_i| \leq 3! = 6$. Thus, for $i \in [3]$ we obtain that $w_i = O(1/\epsilon)$. All in all, we get $\|w\|_2 = O(1/\epsilon)$, and thus $\gamma(h^*) = \Omega(\epsilon)$. Applying Theorem 4, we obtain that ALuMA classifies all the points correctly using $O(\log(1/\gamma(h^*))) \cdot \text{OPT}_{\max} = O(\log^2(1/\epsilon))$ labels.

Part III CAL examines the examples sequentially at a random order, and queries the label of any point whose label is not determined by previous examples. Thus, if the true hypothesis is all-positive on S_a , and CAL sees all the points in S_a before seeing any point in S_b , it will request $\Omega(1/\epsilon)$ labels. Hence, it suffices to show that there is a large probability that CAL will indeed examine all of S_a before examining any point from S_b . Let A be the event that the first $\frac{1}{\epsilon} \log \frac{4}{\epsilon}$ examples of an i.i.d. sample contain any element from S_b . Then, by the union bound, $\mathbb{P}(A) \leq \frac{1}{\epsilon} \log \left(\frac{4}{\epsilon}\right) \cdot \frac{\epsilon}{4 \log \frac{4}{\epsilon}} = 1/4$.

Assume now that A does not occur. Let B be the event that the first $\frac{1}{\epsilon} \log \frac{1}{\epsilon}$ examples do not contain all the elements in S_a . Then, by the union bound, $P(B) \leq \frac{1}{\epsilon} (1 - \epsilon)^{\frac{1}{\epsilon} \log \frac{4}{\epsilon}} \leq 1/4$. All in all, with probability at least $1/2$, CAL see all the points in S_a before seeing any point in S_b and thus its label complexity is $\Omega(1/\epsilon)$. ■

Proof [of Lemma 10] We prove the lemma for the case $h^*(a_{1/2\epsilon}) = 1$. The case $h^*(a_0) = -1$ can be proved similarly. Let w^* be any hyperplane which is consistent with h^* . Let $n_1 = \frac{1}{4\epsilon} - \frac{1}{2}$ and let

$n_2 = n_1 + 1$. Then

$$\begin{aligned} b_{n_1} &= (\cos(\pi/2 - \pi\epsilon), \sin(\pi/2 - \pi\epsilon), 0), \text{ and} \\ b_{n_2} &= (\cos(\pi/2 + \pi\epsilon), \sin(\pi/2 + \pi\epsilon), 0). \end{aligned}$$

By the assumption of the lemma, $\langle w^*, b_{n_1} \rangle > 0$ and $\langle w^*, b_{n_2} \rangle < 0$. It follows that $w^*[1] \sin \pi\epsilon > w^*[2] \cos \pi\epsilon$ and $-w^*[1] \sin \pi\epsilon < w^*[2] \cos \pi\epsilon$. As a consequence, we obtain that $|w^*[2]| < w^*[1] \tan(\pi\epsilon)$.

Now, choose some $n \in \{0, \dots, 1/\epsilon - 1\}$. We show that the corresponding element in S_a is labeled positively. First, from the last inequality, we obtain

$$\begin{aligned} \langle w^*, a_n \rangle &= \frac{1}{\sqrt{2}} \langle w^*, (\cos 2\pi\epsilon n, \sin 2\pi\epsilon n, 1) \rangle \\ &\geq \frac{1}{\sqrt{2}} (w_1^* (\cos 2\pi\epsilon n - \tan(\pi\epsilon) \sin(2\pi\epsilon n)) + w_3^*). \end{aligned} \quad (2)$$

We will now show that

$$\forall n \in \{0, 1, \dots, 1/\epsilon - 1\}, \quad \cos 2\pi\epsilon n - \tan(\pi\epsilon) \sin(2\pi\epsilon n) \geq -1. \quad (3)$$

From symmetry, it suffices to prove this for every $n \in \{0, 1, \dots, 1/(2\epsilon) - 1\}$. We divide our range and conclude for each part separately; since $\epsilon < 1/8$, we have that $\tan \epsilon\pi < 1$. Then, $\cos \alpha - \tan(\pi\epsilon) \sin \alpha \geq -1$ in the range $\alpha \in [0, \pi/2]$. For $\alpha \in [\pi/2, \pi - \pi\epsilon]$, it can be shown that the function $\cos \alpha - \tan(\pi\epsilon) \sin \alpha$ is monotonically decreasing, thus it suffices to show that the inequality holds for $n = 1/(2\epsilon) - 1$. Indeed,

$$\begin{aligned} \cos(\pi - 2\epsilon\pi) - \tan(\epsilon\pi) \sin(\pi - 2\epsilon\pi) &= -\cos(2\pi\epsilon) - 2\sin^2(\pi\epsilon) \\ &= -\cos^2(\pi\epsilon) - \sin^2(\pi\epsilon) \\ &= -1. \end{aligned}$$

Therefore, we obtain from Equation (2) and Equation (3) that

$$\frac{1}{\sqrt{2}} \langle w^*, a_n \rangle \geq \frac{1}{\sqrt{2}} (-w^*[1] + w^*[3]) = \langle w^*, (-1/\sqrt{2}, 0, 1/\sqrt{2}) \rangle = \langle w^*, a_{1/2\epsilon} \rangle > 0,$$

where the last inequality follows from the assumption that $h^*(a_{1/2\epsilon}) = 1$. ■

B Randomization in the Optimal Algorithm

Recall that ALuMA is allowed to use randomization, and it can fail to output the correct label with probability δ . In contrast, in the definition of OPT_{\max} we required that the optimal algorithm always succeeds, in effect making it deterministic. One may suggest that the approximation factor we achieve for ALuMA in Theorem 4 is due to this seeming advantage for ALuMA. We now show that this is not the case—the same approximation factor can be achieved when ALuMA and the optimal algorithm are allowed the same probability of failure. Let m be the size of the pool and let d be the dimension of the examples, and set $\delta_0 = \frac{1}{2m^d}$. Denote by $N_\delta(\mathcal{A}, h)$ the number of calls to L that \mathcal{A} makes before outputting $(L(x_1), \dots, L(x_m))$ with probability at least $1 - \delta$, for $L \Leftarrow h$. Define $\text{OPT}_{\delta_0} = \min_{\mathcal{A}} \max_h N_{\delta_0}(\mathcal{A}, h)$.

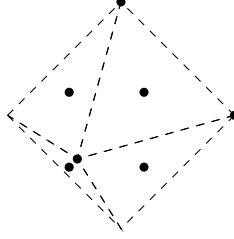
First, note that by setting $\delta = \delta_0$ in ALuMA, we get that $N_\delta(\text{ALuMA}, h) \leq O(\log(1/P(h)) \cdot \text{OPT}_{\max})$. Moreover, ALuMA with $\delta = \delta_0$ is polynomial in m and d (since it is polynomial in $\ln(1/\delta)$). Second, by Sauer's lemma there are at most m^d different possible labelings for the given pool. Thus by the union bound, there exists a fixed choice of the random bits used by an algorithm that achieves OPT_{δ_0} , that leads to the correct identification of the labeling for *all* possible labelings $L(1), \dots, L(m)$. It follows that $\text{OPT}_{\delta_0} = \text{OPT}_{\max}$. Therefore the same factor of approximation can be achieved for ALuMA with $\delta = \delta_0$, compared to OPT_{δ_0} .

C Example: Gap between OPT and CAL

Example 3 Consider a distribution in \mathbb{R}^d that is supported by two types of points on an octahedron (see an illustration for \mathbb{R}^3 below).

1. Vertices: $\{e_1, \dots, e_d\}$.
2. Face centers: z/d for $z \in \{-1, +1\}^d$.

Consider the hypothesis class $\mathcal{W} = \{x \mapsto \text{sgn}(\langle x, w \rangle - 1 + \frac{1}{d}) \mid w \in \{-1, +1\}^d\}$. Each hypothesis in \mathcal{W} , defined by some $w \in \{-1, +1\}^d$, classifies at most $d + 1$ data points as positive: these are the vertices e_i for i such that $w[i] = +1$, and the face center w/d .



Theorem 11 Consider Example 3 for $d \geq 3$, and assume that the pool of examples includes the entire support of the distribution. There is an efficient algorithm that finds the correct hypothesis from \mathcal{W} with at most d labels. On the other hand, with probability at least $\frac{1}{e}$ over the randomization of the sample, CAL uses at least $\frac{2^d + d}{2d + 3}$ labels to find the correct separator.

Proof First, it is easy to see that if $h^* \in \mathcal{W}$ is the correct hypothesis, then

$$w = (h^*(e_1), \dots, h^*(e_d)).$$

Thus, it suffices to query the d vertices to discover the true w .

We now show that the number of queries CAL asks until finding the correct separator is exponential in d . CAL inspects the unlabeled examples sequentially, and queries any example whose label cannot be inferred from previous labels. Consider some run of CAL (determined by the random ordering of the sample). Assume w.l.o.g. that each data point appears once in the sample. Let S be the set that includes the positive face center and all the vertices. Note that CAL cannot terminate before either querying all the $2^d - 1$ negative face centers, or querying at least one example from S . Moreover, CAL will query all the face centers it encounters before encountering the first example from S . At each iteration t before encountering an example from S , there is a probability of $\frac{d+1}{2^d + d - t}$ that the next example is from S . Therefore, the probability that the first $T = \frac{2^d + d}{2d + 3}$ examples are not from S is

$$\prod_{t=0}^{T-1} \left(1 - \frac{d+1}{2^d + d - t}\right) \geq \left(1 - \frac{d+1}{2^d + d - T}\right)^T \geq e^{-2T \frac{d+1}{2^d + d - T}} = e^{\frac{-2(d+1)}{\frac{2^d + d}{2d + 3} - 1}} = \frac{1}{e},$$

where in the second equality we used $1 - a \geq \exp(-2a)$ which holds for all $a \in [0, \frac{1}{2}]$. Therefore, with probability at least $\frac{1}{e}$ the number of queries is at least $\frac{2^d + d}{2d + 3}$. ■

D Additional Experiments

In this appendix we provide the details on our implementation of the algorithms in our experiments. We also provide results of additional experiments comparing ALuMA, TK, CAL, QBC and passive ERM. Our implementation of ALuMA uses hit-and-run samples instead of full-blown volume estimation. QBC is also implemented using hit-and-run as in Gilad-Bachrach et al. [2005]. For both ALuMA and QBC, we used a fixed number of mixing iterations for hit-and-run, which we set to

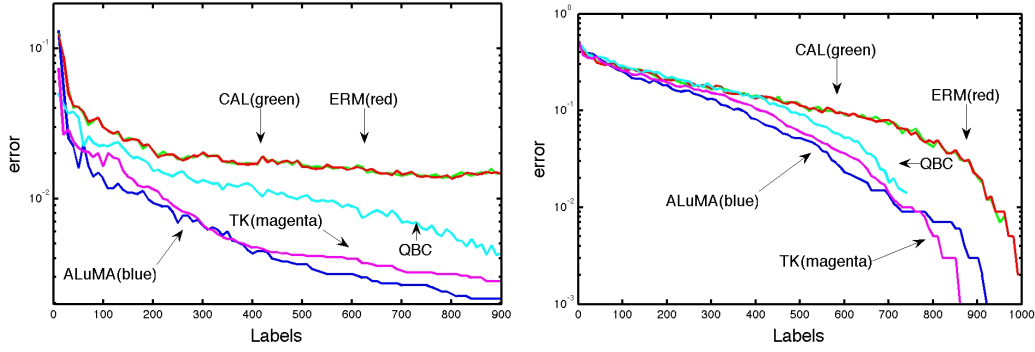


Figure 4: Left: MNIST 4 vs. 7. Right: PCMAC

1000. We also fixed the number of sampled hypotheses at each iteration of ALuMA to 1000, and used the same set of hypotheses to calculate the majority vote for classification. CAL and QBC examine the examples sequentially, thus the input provided to them was a random ordering of the example pool. The algorithm TK is the first heuristic proposed in Tong and Koller [2002], in which the example chosen at each iteration is the one closest to the max-margin solution of the labeled examples known so far. Since the active learners operate by reducing the training error, the graphs below compare the training errors of the different algorithms. The test errors show a similar trend.

In each of the algorithms, the classification of the training examples is done using the version space defined by the queried labels. The theory for CAL and ERM allows selecting an arbitrary predictor out of the version space. In QBC, the hypothesis should be drawn uniformly at random from the version space. We have found that all the algorithms show a significant improvement in classification error if they classify using the majority vote classification proposed for ALuMA. Therefore, in all of our experiments below, the results for all the algorithms are based on a majority vote classification.

Our first data set is MNIST⁵. The examples in this data set are gray-scale images of handwritten digits in dimension 784. Each digit has about 6,000 training examples. We performed binary active learning by pre-selecting pairs of digits. One experiment was already reported in Section 6.2. The second experiments is reported in Figure 4 (left): In this case we trained the active learners on data of the digits 4 and 7, which are linearly separable just like 3 and 5. The results are very similar to those obtained for the digits 3 and 5.

We also tested the algorithms on the PCMAC dataset⁶. This is a real-world data set, which represents a two-class categorization of the 20-Newsgroup collection. The examples are web-posts represented using bag-of-words. The original dimension of examples is 7511. We used the Johnson-Lindenstrauss projection to reduce the dimension to 300, which kept the data still separable. We used a training set of 1000 examples. Figure 4 (right) depicts the results. We were not able to run QBC long enough to use its entire label budget, as it tends to become slower when the training error becomes small.

For the uniform distribution, Figure 5 depicts the training error as a function of the label budget when learning a random halfspace over the uniform distribution in \mathbb{R}^{10} . The difference between the performance of the different algorithms is less marked for $d = 10$ than for $d = 100$ (see Section 6.2), suggesting that the difference grows with the dimension.

d	ALuMA	TK	QBC	CAL	ERM
10	29	156	50	308	1008
12	38	735	113	862	3958
15	55	959	150	2401	> 20000

Table 1: Octahedron: number of queries to achieve zero error

⁵<http://yann.lecun.com/exdb/mnist/>

⁶<http://vikas.sindhwani.org/datasets/lsm/matlab/pcmac.mat>

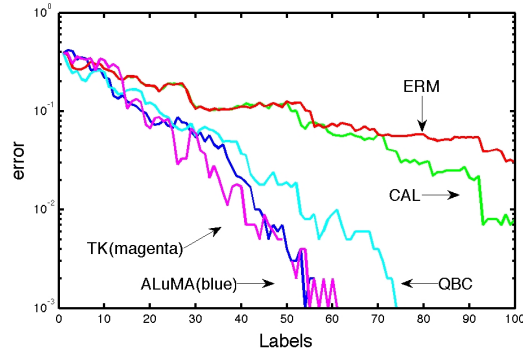


Figure 5: Uniform distribution ($d = 10$).

Finally, we report a synthetic experiment. In this experiment the pool of examples is taken to be the support of the distribution described in Example 3, with an additional dimension to account for halfspaces with a bias. We also added the negative vertices $-e_i$ to the pool. Similarly to the proof of Theorem 11, it suffices to query the vertices to reach zero error. Table 1 lists the number of iterations required in practice to achieve zero error by each of the algorithms. In this experiment, unlike the rest, ALuMA is not only much better than QBC and CAL, it is also much better than TK, which is worse even than QBC here. This suggests that TK might not have guarantees similar to those of ALuMA, despite the fact that they both attempt to minimize the same objective. The number of queries ALuMA requires is indeed close to the number of vertices.